Chương 5
Con trỏ, mảng, chuỗi ký tự

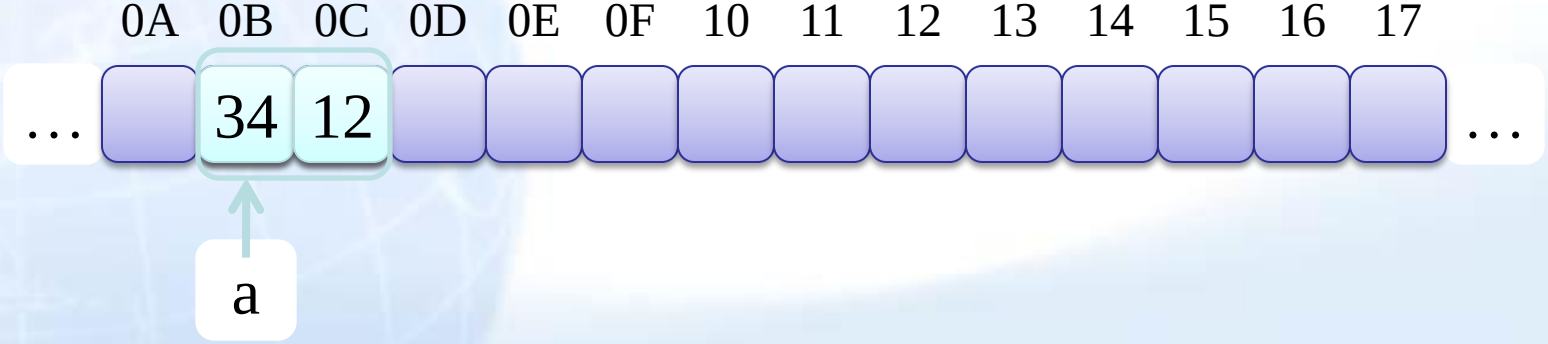
- Con trỏ
- Mảng một chiều
- Mảng hai chiều
- Chuỗi ký tự

Kiến trúc máy tính

- Bộ nhớ máy tính
  - *Bộ nhớ RAM chứa rất nhiều ô nhớ, mỗi ô nhớ có kích thước 1 byte.*
  - *RAM dùng để chứa một phần hệ điều hành, các lệnh chương trình, các dữ liệu…*
  - *Mỗi ô nhớ có địa chỉ duy nhất và địa chỉ này được đánh số từ 0 trở đi.*
  - *Ví dụ*
    - RAM 512MB được đánh địa chỉ từ 0 đến $2^{29} - 1$
    - RAM 2GB được đánh địa chỉ từ 0 đến $2^{31} - 1$

Khai báo biến trong C

- Quy trình xử lý của trình biên dịch
  - *Dành riêng một vùng nhớ với địa chỉ duy nhất để lưu biến đó.*
  - *Liên kết địa chỉ ô nhớ đó với tên biến.*
  - *Khi gọi tên biến, nó sẽ truy xuất tự động đến ô nhớ đã liên kết với tên biến.*
- Ví dụ: int a = 0x1234;// Giả sử địa chỉ 0x0B



Con trỏ (pointer)

- **Khái niệm:** Là **một biến** dùng để lưu địa chỉ của một biến, mỗi loại địa chỉ sẽ có một kiểu con trỏ tương ứng (phụ thuộc vào loại dữ liệu lưu trữ trong địa chỉ đó)
- Kích thước của biến con trỏ luôn là 2 byte.
- **Các loại con trỏ**

  Con trỏ kiểu int dùng để chứa địa chỉ của các biến kiểu int. Tương tự ta có con trỏ kiểu float, double, …

Con trỏ (pointer)

❖ **Cách khai báo con trỏ**

```
Kiểu dữ liệu  * TênConTrỏ;
```

***Ý nghĩa*:** Khai báo một biến có tên là TênConTrỏ dùng để chứa địa chỉ của các biến có kiểu Kiểu dữ liệu.

❖ Ví dụ:

```
int *px, y;
float *pm;
```

Con trỏ (pointer)

❖ **Gán địa chỉ của biến cho biến con trỏ**

```
TênConTrỏ = &TênBiến
```

***Ý nghĩa*:** Dùng & để lấy ra địa chỉ bộ nhớ (memory address) của 1 biến

Ví dụ:

```
int a=6;
int* c= &a;  // &a là địa chỉ bộ nhớ của biến a
```

Con trỏ (pointer)

❖ Cách lấy giá trị của con trỏ

```
* TênConTrỏ
```

***Ý nghĩa*:** Dùng * để truy cập (access) đến nội dung (content) của biến mà 1 con trỏ đang chỉ đến

```
int a=6;
int *c= &a;
*c=7;       /*Thay đổi nội dung của biến a bằng cách
              dùng địa chỉ của nó được chứa trong con trỏ c*/
```

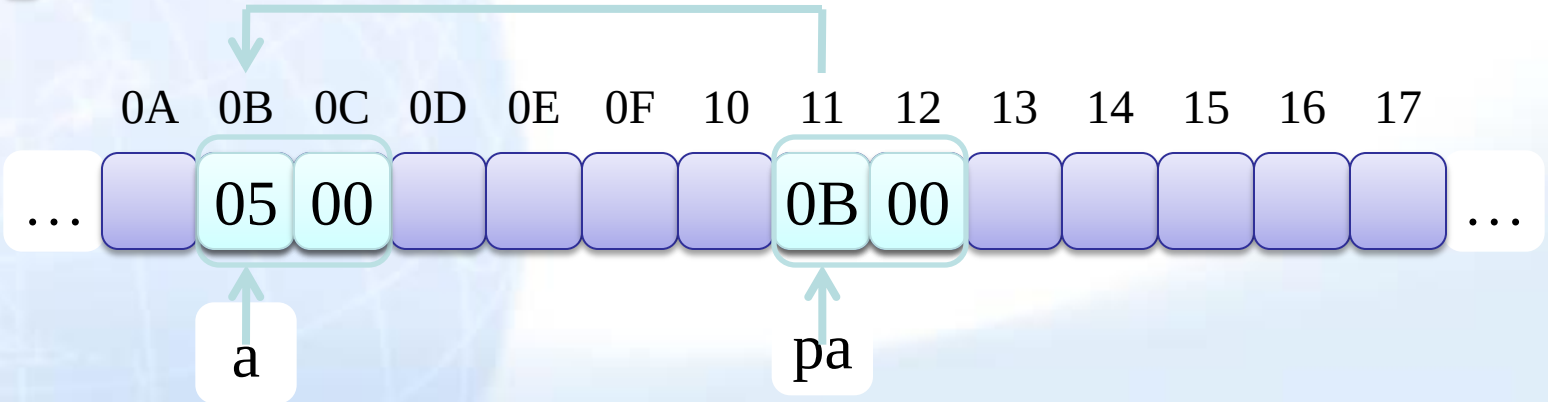
tương đương với

```
a=7;
```

Sử dụng con trỏ

- Truy xuất đến ô nhớ mà con trỏ trỏ đến
  - *Con trỏ chứa một số nguyên chỉ địa chỉ.*
  - *Vùng nhớ mà nó trỏ đến, sử dụng toán tử* *
- Ví dụ

```
int a = 5, *pa = &a;
printf("%d\n", pa);  // Giá trị biến pa
printf("%d\n", *pa); // Giá trị vùng nhớ pa trỏ đến
printf("%d\n", &pa); // Địa chỉ biến pa
```



Ví dụ

```
void main()
{
        int *p, *q;
        int x = 5, z;
        p = &x;
        q = p;
        z = *p + 3*(*q);
        printf("Gia tri cua z = %d",z);
}
```

Kết quả

Cấp phát vùng nhớ cho con trỏ

- **Có 2 cách để dùng được biến con trỏ**
  1. Cho nó chứa địa chỉ của 1 vùng nhớ đang tồn tại

```
int a=6;
int* c;
c= &a;  // &a là địa chỉ bộ nhớ của biến a
```

  2. Cấp phát 1 vùng nhớ mới, rồi cho con trỏ chỉ đến

```
int * ptr;
ptr = (int*)malloc(sizeof(int));
*ptr=6;
```

Cấp phát vùng nhớ cho con trỏ

- **void *malloc(size_t size)** Cấp phát vùng nhớ có kích thước là size (byte)
- **void *calloc(size_t nitems, size_t size)** Cấp phát vùng nhớ có kích thước là nitems*size (byte)
- **Ví dụ:**

```
int a, *pa, *pb;
pa = (int*)malloc(sizeof(int)); /* Cấp phát vùng nhớ có kích thước bằng với kích thước của một số nguyên */
pb= (int*)calloc(10, sizeof(int)); /* Cấp phát vùng nhớ có thể chứa được 10 số nguyên*/
pb= new int;  //Cấp pháp vùng nhớ trong C++
```

Giải phóng vùng nhớ cho con trỏ

- **void free(void *block)** Giải phóng vùng nhớ được quản lý bởi con trỏ block
- **Ví dụ**

free(pa);

free(pb);

=> giải phóng vùng nhớ do 2 biến con trỏ pa và pb đang chỉ đến

❖ Giải phóng vùng nhớ con trỏ trong C++:

delete ten_con_tro;

Ví dụ: delete pa;

Gán NULL cho 1 con trỏ

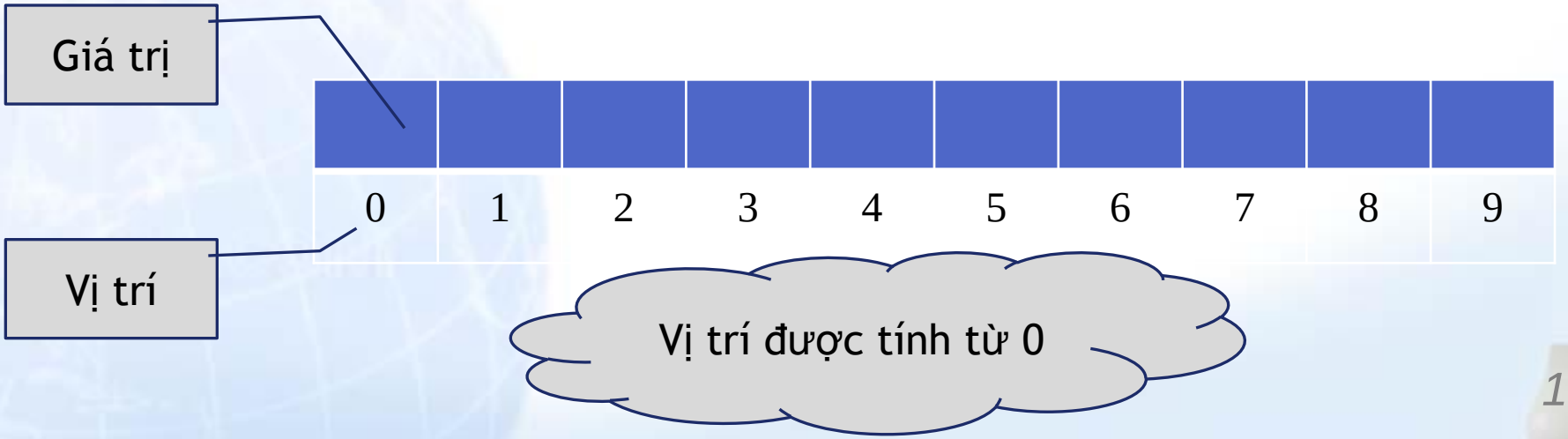
❖**Ví dụ:**

```
int x=25;
int *ptr;
ptr=&x;
ptr=NULL;
```

❖Lệnh gán **ptr=NULL** => cho con trỏ ptr không trỏ vào (không chứa địa chỉ) vùng nhớ nào cả

Kiểu mảng

- Mảng thực chất là một biến được cấp phát bộ nhớ liên tục và bao gồm nhiều biến thành phần.
- Các thành phần của mảng là tập hợp các biến có cùng kiểu dữ liệu và cùng tên. Do đó để truy xuất các biến thành phần, ta dùng cơ chế chỉ mục.



Kiểu mảng

- Ta có thể chia mảng làm 2 loại:
  - *Mảng 1 chiều*
  - *Mảng nhiều chiều*

Mảng 1 chiều

❖*Khai báo mảng với số phần tử xác định*

- ***Cú pháp:***

< Kiểu dữ liệu > < Tên mảng > [< Số phần tử tối đa của mảng>];

- ***Ví dụ:***

int a[100]; //Khai bao mang so nguyen a gom 100 phan tu

float b[50]; //Khai bao mang so thuc b gom 50 phan tu

char str[30]; //Khai bao mang ky tu str gom 30 ky tu

Nhằm thuận tiện cho việc viết chương trình, ta nên định nghĩa hằng số MAX ở đầu chương trình – là kích thước tối đa của mảng - như sau:

```
#define MAX 100
void main()
{
        int a[MAX], b[MAX];
        //Các lệnh
}
```

Khai báo và gán giá trị ban đầu cho mảng

❖Khai báo và gán từng phần tử

int a[5] = {3, 6, 8, 1, 12};

Giá trị	3	6	8	1	12
Vị trí	*0*	*1*	*2*	*3*	*4*

Gán toàn bộ phần tử có cùng giá trị

int a[8] = {3};

Giá trị	3	3	3	3	3	3	3	3
Vị trí	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*

Truy xuất giá trị mảng

❖ Cú pháp

TênMảng [vị trí cần truy xuất]

❖ Ví dụ:

```
void main()
{
        int a[5] = {3, 6, 8, 11, 12};
        printf("Gia tri mang tai vi tri 3 = %d",a[3]);
}
```

Kết quả: Giá trị mảng tại vị trí 3 = 11

Chú ý: Các chỉ số được đánh số từ **0 1 2 3 4**

Các thao tác trên mảng

- Nhập
- Xuất (liệt kê)
- Tìm kiếm
- Đếm
- Sắp xếp
- Kiểm tra mảng thỏa điều kiện cho trước
- Tách/ ghép mảng
- Chèn / xóa

Nhập xuất mảng

Sự tương quan mảng và con trỏ

- Khi khai báo một mảng thì **tên của mảng là một hằng địa chỉ**, chứa địa chỉ của phần tử đầu tiên (phần tử có chỉ số 0).
- Xét khai báo: int a[5]; int *pa = a; khi đó con trỏ pa cũng giữ địa chỉ của phần tử đầu tiên của mảng a và pa+i (hoặc pa[i]) là địa chỉ của phần tử a[i].

Sự tương quan mảng và con trỏ

- Các khai báo tương đương
  - `int *pa;` ⇔ `int pa[];`
  - `double *pa;` ⇔ `double pa[];`
  - `char *pa;` ⇔ `char pa[];`
  - `long *pa;` ⇔ `long pa[];`

Khai báo mảng bằng con trỏ

- **Cú pháp:**

  < Kiểu dữ liệu > *< Tên mảng >;

- **Ví dụ :**

  int *p; // khai bao con tro p

  int b[100];

  p = (int*)malloc(sizeof(int)*100); //C++ p = new int[100];

  p = b; // p tro vao phan tu 0 cua mang b

- Với cách viết như trên thì ta có thể hiểu các cách viết sau là tương đương

  p[i] ⇔ *(p + i) ⇔ b[i] ⇔ *(b+i)

- Cấp phát: **hàm malloc** ***(C++ new)***
- Giải phóng **free(p)** **(C++ delete p)**

Mảng và hàm

- **Khai báo hàm nhập mảng**

```
void NhapMang(int a[], int &n);
```

Phân tích:

- *Tên hàm:* **`NhapMang`**
- *Tham số* **`n`** *là tham chiếu.*
- *Tham số* **`a`** *là tham trị vì* **`a`** *là con trỏ hằng.*
- *Giá trị trả về: không trả về giá trị cụ thể.*

- ***Khai báo hàm xuất mảng***

```
void XuatMang(int a[], int n);
```

Viết chương trình nhập xuất mảng bằng hàm????

Mảng và hàm

Chương trình nhập xuất mảng bằng hàm

Mảng và hàm

Nhập mảng ngẫu nhiên

➢ Đối với Bordland C++:

- Bước 1: khai báo thư viện
  #include <stdlib.h>
- Bước 2: Khởi động bộ tạo random ngoài vòng lặp
  randomize();
- Bước 3: lấy số random
  x = random(n);

//Hàm random(n) cho kết quả là 1 số ngẫu nhiên có trị 0 đến n -1

=> muốn lấy cái giá trị âm thì sao?????

Nhập mảng ngẫu nhiên

➢ ĐỐI VỚI VISUAL C++ hoặc DevC++

❖ Bước 1: Khai báo thư viện

```
#include <time.h>
#include <stdlib.h>
```

❖ Bước 2: Đặt điểm bắt đầu cho việc sinh số ngẫu nhiên của hàm rand()

```
srand(time(NULL));  //đặt bên ngoài vòng lặp lấy ngẫu nhiên
```

❖ Bước 3: Lấy số random

```
x = rand()%n;
```

//hàm rand sẽ lấy số ngẫu nhiêu trong khoảng [0, RAND_MAX]

=> rand()% n lấy số ngẫu nhiên khoảng 0 đến n-1

Nếu muốn số ngẫu nhiên trong đoạn [0,n] thì sao?

Sinh số nằm trong khoảng từ [m, n] thì sao ?

Viết lại hàm nhập mảng ????

Nhập mảng ngẫu nhiên Bordland C++

Nhập mảng ngẫu nhiên VISUAL C++

Một số thuật toán

1. <u>Tính tổng/tích các phần tử mảng:</u> Duyệt tòan bộ mảng, thực hiện cộng hoặc nhân tích lũy

2. <u>Tìm kiếm</u>: Duyệt mảng cho đến khi tìm thấy.

Hàm Tìm Kiếm (dùng for)

Một số thuật toán

3. <u>Liệt kê các phần tử chẵn:</u> Duyệt toàn bộ mảng, xuất ra các phần tử chẵn

Một số thuật toán

4. <u>Đếm các phần tử dương trong mảng:</u> Duyệt toàn bộ mảng, đếm các phần tử dương

5. Xây dựng hàm kiểm tra số nguyên tố và hàm đếm các phần tử mảng là số nguyên tố ???? Về nhà làm

Hàm void main

Hàm void main

Tìm giá trị lớn nhất của mảng

- Yêu cầu
  - *Cho trước mảng a có n phần tử. Tìm giá trị lớn nhất trong a (gọi là max)*
- Ý tưởng
  - *Giả sử giá trị max hiện tại là giá trị phần tử đầu tiên a[0]*
  - *Lần lượt kiểm tra các phần tử còn lại để cập nhật max.*

Hàm tìm Max

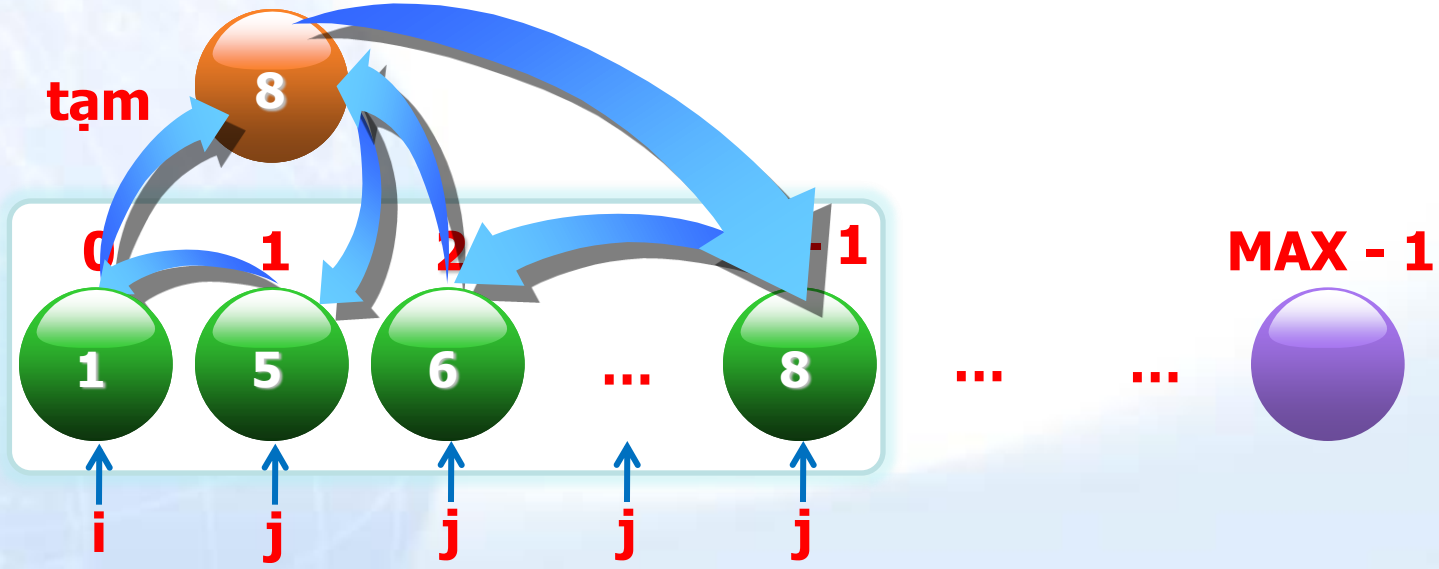
Sắp xếp mảng thành tăng dần

❖ Yêu cầu

- *Cho trước mảng a kích thước n. Hãy sắp xếp mảng a đó sao cho các phần tử có giá trị tăng dần.*

❖ Ý tưởng

- *Sử dụng 2 biến i và j để so sánh tất cả cặp phần tử với nhau và hoán vị các cặp nghịch thế (sai thứ tự).*



Sắp Xếp Tăng

Mẫu phương thức sắp thứ tự tăng:

Mảng nhiều chiều

- Mảng nhiều chiều là mảng có từ 2 chiều trở lên.
- Điều đó có nghĩa là mỗi phần tử của mảng là một mảng khác.
- Người ta thường sử dụng mảng nhiều chiều để lưu các ma trận, các tọa độ 2 chiều, 3 chiều…

Khai báo mảng 2 chiều

❖ Khai báo

<KiểuDL> <tênbiếnmảng> [<sốphầntửhàng>][<sốphầntửcột>];

Ví dụ: float m[8][9]; // mảng 2 chiều có 8*9 phần tử là số thực

int a[3][4]; // mảng 2 chiều có 3*4 phần tử là số nguyên

❖ Truy xuất phần tử mảng 2 chiều

Tênmảng[Chỉ số dòng][Chỉ số cột]

Ví dụ: int a[3][4] = { {2,3,9,4} , {5,6,7,6} , {2,9,4,7} };

Với các khai báo như trên ta có:

a[0][0] = 2; a[0][1] = 3;

a[1][1] = 6; a[1][3] = 6;

	0	1	2	3
0	2	3	9	4
1	5	6	7	6
2	2	9	4	7

Nhập xuất mảng 2 chiều

- *Nhập:*

```
int a[4][5], i, j;
for (i = 0; i<4; i++)
    for (j = 0; j<5; j++)
    {  // phần tử đầu tiên là a[0][0]
       printf("a[%d][%d]=", i, j);
       scanf("%d", &a[i][j]);
    }
```

- *Xuất:*

```
for (i = 0; i<4; i++)
{
    for (j = 0; j<5; j++)
       printf("%5d", a[i][j]);
    printf("\n");
}
```

Mảng hai chiều

❖ **Khai báo qua con trỏ**

< Kiểu dữ liệu > **<Tên mảng>;

Ví dụ :

```
int **A ;       // Khai báo mảng động 2 chiều kiểu int
float **B ;     // Khai báo mảng động 2 chiều kiểu float
A = new int*[6];  //Cấp phát bộ nhớ cho số dòng của ma trận A
for(int i = 0; i<6; i++)
    A[i] = new int[10]; //Cấp phát bộ nhớ cho các phần tử của mỗi dòng
```

❖ **Truyền mảng 2 chiều cho hàm**

- ***Hàm bị gọi ví dụ:***

```
void ABC(int a[ ][100], int n, int m)   //phải cho biết số cột tối đa
```

- ***Gọi hàm:***

```
int a[100][100], x, y;
ABC(a, x, y);
```

Nhập xuất mảng 2 chiều

Nhập xuất mảng 2 chiều

Kĩ thuật đặt lính canh

❖ ***Viết hàm tìm phần tử nhỏ nhất trong ma trận***

Định nghĩa kiểu dữ liệu mới

```
#define MAX 100
typedef <kieudulieu> MATRAN[MAX][MAX];
```

Ví dụ : Khai báo ma trận các số nguyên a.

```
#define MAX 100
typedef int MATRAN[MAX][MAX];
MATRAN a;  // int a[MAX][MAX];
```

Chuỗi ký tự

- Chuỗi ký tự là một dãy các phần tử, mỗi phần tử có kiểu ký tự
- Trong ngôn ngữ C, chuỗi ký tự là một dãy các ký tự đặt trong hai dấu nháy kép.
- Khi gặp chuỗi ký tự, máy sẽ cấp phát khoảng nhớ cho 1 mảng kiểu char đủ lớn để chứa các ký tự xâu và '\0'
- Chuỗi rỗng được ký hiệu bằng hai dấu nháy kép đi liền nhau : ""
- *Chú ý: Cần phân biệt mảng các ký tự và chuỗi ký tự. Đối với chuỗi ký tự, ký tự kết thúc chuỗi là '\0'*

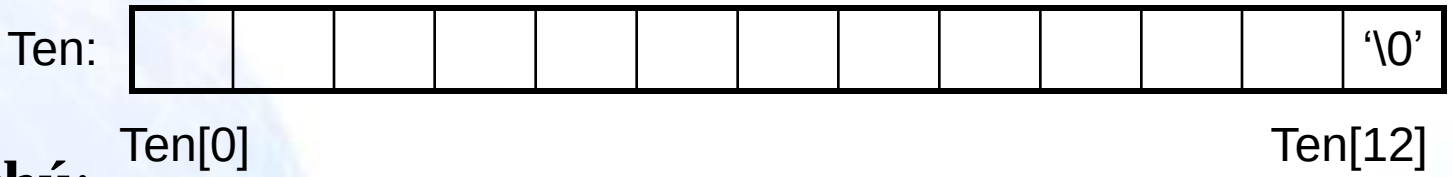
Khai báo theo mảng

❖ **Cú pháp:**

char Tênchuỗi[<Số ký tự tối đa của chuỗi>];

❖ **Ví dụ:** **char** Ten[13];

=> bộ nhớ sẽ cung cấp 13 bytes để lưu trữ nội dung của chuỗi ký tự Ten;
byte cuối cùng lưu trữ ký tự '\0' để chấm dứt chuỗi



❖ **Ghi chú**:

- Chiều dài tối đa của biến chuỗi: 1..255 bytes.
- Chuỗi ký tự được kết thúc bằng ký tự '\0' =>khai báo độ dài của chuỗi luôn luôn khai báo dư 1 phần tử để chứa ký tự '\0'

Khai báo theo con trỏ

- **Cú pháp:** **char *<Biến>;**
- **Ví dụ:** char *Ten;

- Trong khai báo này, bộ nhớ sẽ dành 2 byte để lưu trữ địa chỉ của biến con trỏ Ten đang chỉ đến.
- Chưa cung cấp nơi để lưu trữ dữ liệu.
- Do đó phải cấp phát vùng nhớ bằng hàm malloc hoặc calloc trong "alloc.h" hoặc "stdlib.h"
- Ví dụ:

```
char *Ten;
Ten = (char*)malloc(20*sizeof(char));
//hoặc Ten = "chuoi nao do" //?????
```

Các hàm nhập xuất chuỗi

- Hàm nhập chuỗi: **gets**
  Ví dụ: gets(hoten);

  Hàm tự động thêm ký tự NULL (‘\0’) vào cuối biến chuỗi.

- Hàm xuất chuỗi: **puts**
  Ví dụ: puts(hoten);

  - ✓ *Hàm **scanf**?*
  - ✓ *Hàm **printf** với mã định dạng là %s*

  *Chú ý: Khi dùng hàm nhập chuỗi sau hàm scanf phải sử dụng hàm **fflush (stdin) trước** để khử ký tự ‘\n’ vì ký tự này làm trôi hàm gets…*

Ví dụ 1

Ví dụ 2

Truy xuất chuỗi

Cách 1. Truy xuất giống mảng ký tự.

Ví dụ:

```
char s[]={‘T’,’h’,’u’,’\0’};
int i=0;
while (s[i]!=‘\0’)
{
        printf(“%c”,s[i]);
        i++;
}
```

Cách 2. Sử dụng hàm chuỗi.

Các hàm thư viện – <string.h>

❖Tính độ dài của chuỗi s

```
int strlen(char *s);
```

```
void main()
{
    char *s = "Lap trinh C";
    printf("Do dai s = %d",strlen(s));
}
```

Kết quả:

Do dai s = 11

Các hàm thư viện – <string.h>

- **Sao chép** nội dung chuỗi nguồn vào chuỗi đích, nội dung của chuỗi đích sẽ bị xóa

```
strcpy(char *đích, char *nguồn);
```

- Chép n ký tự từ chuỗi nguồn sang chuỗi đích. Nếu chiều dài nguồn < n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào đích

```
strncpy(char *đích, char *nguồn, int n);
```

Các hàm thư viện – <string.h>

- **Nối chuỗi** s2 vào chuỗi s1

  strcat(char *s1, char *s2);

- Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1

  strncat(char *s1, char *s2, int n);

- **So sánh 2 chuỗi** s1 và s2 theo nguyên tắc thứ tự từ điển. ***Phân biệt chữ hoa và thường.*** Trả về:

  0: nếu s1 bằng s2.

  1: nếu s1 lớn hơn s2.

  -1: nếu s1 nhỏ hơn s2.

  int strcmp(char *s1, char *s2);

Các hàm thư viện – <string.h>

- So sánh n ký tự đầu tiên của s1 và s2, giá trị trả về tương tự hàm strcmp()

  int strncmp(char *s1,char *s2, int n);

- So sánh chuỗi s1 và s2 nhưng không phân biệt hoa thường, giá trị trả về tương tự hàm strcmp()

  int stricmp(char *s1,char *s2);

- So sánh n ký tự đầu tiên của s1 và s2 nhưng không phân biệt hoa thường, giá trị trả về tương tự hàm strcmp()

  int strnicmp(char *s1,char *s2, int n);

- Tìm sự xuất hiện đầu tiên của ký tự c trong chuỗi s. Trả về:
  - NULL: nếu không có
  - Địa chỉ c: nếu tìm thấy

char *strchr(char *s, char c);

- Tìm sự xuất hiện đầu tiên của chuỗi s2 trong chuỗi s1. Trả về:
  - NULL: nếu không có
  - Ngược lại: Địa chỉ bắt đầu chuỗi s2 trong s1

char *strstr(char *s1, char *s2);

Đổi ký tự hoa sang thường và ngược lại

- **Đổi một ký tự thường thành ký tự hoa** (trong ctype.h)

  Cú pháp: char toupper(char c)

- **Đổi chuỗi chữ thường thành chuỗi chữ hoa**

  Cú pháp: char* strupr(char *s)

- **Đổi một ký tự hoa thành ký tự thường** (trong ctype.h)

  Cú pháp: char tolower(char c)

- **Đổi chuỗi chữ hoa thành chuỗi chữ thường**

  Cú pháp: char *strlwr(char *s)

Đổi từ chuỗi ra số - atoi(), atof(), atol() (trong stdlib.h)

- **Cú pháp :**

  int atoi(const char *s) : chuyển chuỗi thành số nguyên

  long atol(const char *s) : chuyển chuỗi thành số nguyên dài

  float atof(const char *s) : chuyển chuỗi thành số thực

- Nếu chuyển đổi không thành công, kết quả trả về của các hàm là 0.

- **Ví dụ:**

  atoi("1234")=> 1234

Ví dụ

Ví dụ chuyển đổi số